

Contents

1 Routine/Function Prologues	2
1.0.1 maketiles_gds() (Source File: maketiles_gds.F90)	2

1 Routine/Function Prologues

1.0.1 maketiles_gds() (Source File: maketiles_gds.F90)

This primary goal of this routine is to determine tile space for GDS based I/O

REVISION HISTORY:

```

1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Major F90 and major structure revision
3 Jan 2000: Minor T=0 bug fix, should have no effect on output
8 Mar 2000: Brian Cosgrove; Initialized FGRD to 0 For Dec Alpha Runs
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask
04 Feb 2001: Jon Gottschalck; Added option to read and use Koster tile space

```

INTERFACE:

```

subroutine maketiles_gds()
#if ( defined OPENDAP )

```

USES:

```

use lisdrv_module, only: lis, grid, glbgindex, tile
use grid_module
use spmdMod

```

CONTENTS:

```

#if ( defined ABSOFT )
    character*15 :: home
    home = '/home/jim/'
#else
    character*2 :: home
    home = './'
#endif

lis%d%gnc = lis%d%lnc
lis%d%gnr = lis%d%lnr

allocate(recvcnts2(0:npes-1))
allocate(displs2(0:npes-1))

lnr = lis%d%gnr / npes
lnr_last = lis%d%gnr - lnr*(npes-1)

displs2(0) = 0
do t = 1, npes-1
    displs2(t) = displs2(t-1) + lnr*lis%d%gnc
enddo

do t = 0, npes-2

```

```

    recvcnts2(t) = lnr*lis%d%gnc
enddo
    recvcnts2(npes-1) = lnr_last*lis%d%gnc

wlon = 1
elon = lis%d%gnc

slat = iam*lnr + 1
if ( iam == npes-1 ) then
    nlat = lis%d%gnr
else
    nlat = (iam+1)*lnr
endif

if ( iam == npes-1 ) then
    lnr = lnr_last
endif

write(cslat, '(i4)') slat
write(cnlat, '(i4)') nlat
write(cwlon, '(i4)') wlon
write(celon, '(i4)') elon
write(ciam, '(i3)') iam

print*,'DBG: maketiles_gds -- cslat ', cslat, ', (', iam, ')'
print*,'DBG: maketiles_gds -- cnlat ', cnlat, ', (', iam, ')'
print*,'DBG: maketiles_gds -- cwlon ', cwlon, ', (', iam, ')'
print*,'DBG: maketiles_gds -- celon ', celon, ', (', iam, ')'
print*,'DBG: maketiles_gds -- ciam ', ciam, ', (', iam, ')'

lis%p%mfile = trim(home)//trim(adjustl(ciam))///'/'//lis%p%mfile
lis%p%vfile = trim(home)//trim(adjustl(ciam))///'/'//lis%p%vfile

allocate(lat(lis%d%gnc,lnr), stat=ierr)
call check_error(ierr,'Error allocating lat.',iam)

allocate(lon(lis%d%gnc,lnr), stat=ierr)
call check_error(ierr,'Error allocating lon.',iam)

allocate(fgrd(lis%d%gnc,lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating fgrd.',iam)

allocate(mask(lis%d%gnc, lnr), stat=ierr)
call check_error(ierr,'Error allocating mask.',iam)

allocate(pveg(lis%d%gnc,lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating pveg.',iam)

```

```

allocate(tsum(lis%d%gnc, lnr), stat=ierr)
call check_error(ierr,'Error allocating tsum.',iam)

allocate(veg(lis%d%gnc, lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating veg.',iam)

tsum = 0.0

nchp = lis%d%glbnch
print*, 'MSG: maketiles -- Retrieving UMD mask file ', &
trim(lis%p%mfile), ' (',iam,')'
call system("opendap_scripts/getmask.pl "//ciam//" //&
trim(lis%p%mfile)//" "// &
cslat//" //cnlat//" //cylon//" //celon)
print*, 'MSG: maketiles -- Reading ',trim(lis%p%mfile), '&
' (',iam,')'
open(30,file=lis%p%mfile,form='unformatted',status='old')
read(30) lat
read(30) lon
read(30) mask
close(30)
print*, 'MSG: maketiles -- Done reading ',trim(lis%p%mfile), '&
' (',iam,')'
!-----
! Select which tile-space veg. info to use (UMD or Koster)
!-----
! if (lis%p%koster .lt. 1) then           ! Use original UMD indexing
print*, 'MSG: maketiles -- Retrieving UMD veg file ', &
trim(lis%p%vfile), ' (',iam,')'
call system("opendap_scripts/getveg.pl "//ciam//" //&
trim(lis%p%vfile)//" "// &
cslat//" //cnlat//" //cylon//" //celon)
print*, 'MSG: maketiles -- Reading ',trim(lis%p%vfile), '&
' (',iam,')'
open(98,file=lis%p%vfile,form='unformatted')
read(98) lat
read(98) lon
do t = 1, lis%p%nt
  read(98) veg(:,:,t)
enddo
do r=1,lnr
  do c=1,lis%d%gnc
    isum=0.0
    do t=1,lis%p%nt
      isum=isum+veg(c,r,t)
    enddo
    do t=1,lis%p%nt
      fgrd(c,r,t)=0.0
    enddo
  enddo
enddo

```

```

        if(isum.gt.0) fgrd(c,r,t)=veg(c,r,t)/isum
    enddo
    enddo
    close(98)
    print*, 'MSG: maketiles -- Done reading ',trim(lis%p%vfile), &
        (' ,iam, ')
! endif
!-----
! Exclude tiles with MINA (minimum tile grid area),
! normalize remaining tiles to 100%
!-----
print*, 'DBG: maketiles -- re-normalizing tiles', (' ,iam, ')
do r=1,lnr
    do c=1,lis%d%gnc
        rsum=0.0
        do t=1,lis%p%nt
            if(fgrd(c,r,t).lt.lis%d%mina)then
                fgrd(c,r,t)=0.0
            endif
            rsum=rsum+fgrd(c,r,t)
        enddo
        if(rsum.gt.0.0) then
            do t=1,lis%p%nt
                if(rsum.gt.0.0)fgrd(c,r,t)=fgrd(c,r,t)/rsum
            enddo

            rsum=0.0
            do t=1,lis%p%nt
                rsum=rsum+fgrd(c,r,t)
            enddo

            if(rsum.lt.0.9999.or.rsum.gt.1.0001)then
                write(*,*) 'error1 in vegetation tiles',rsum,c,r
            endif
        endif
    enddo
enddo
!-----
! Exclude tiles with MAXT (Maximum Tiles per grid),
! normalize remaining tiles to 100%
! Determine the grid predominance order of the tiles
! PVEG(NT) will contain the predominance order of tiles
!-----
print*, 'DBG: maketiles -- excluding MAXT tiles', (' ,iam, ')
do r=1,lnr
    do c=1,lis%d%gnc
        do t=1,lis%p%nt

```

```

        fvt(t)=fgrd(c,r,t)
        pveg(c,r,t)=0
    enddo
    do i=1,lis%p%nt
        max=0.0
        t=0
        do j=1,lis%p%nt
            if(fvt(j).gt.max)then
                if(fgrd(c,r,j).gt.0) then
                    max=fvt(j)
                    t=j
                endif
            endif
        enddo
        if(t.gt.0) then
            pveg(c,r,t)=i
            fvt(t)=-999.0
        endif
    enddo
    enddo
    enddo
!-----
! Impose MAXT Cutoff
!-----
      print*,'DBG: maketiles -- imposing MAXT cut-off',' (',iam,')'

      do r=1,lnr
          do c=1,lis%d%gnc
              rsum=0.0
              do t=1,lis%p%nt
                  if(pveg(c,r,t).lt.1) then
                      fgrd(c,r,t)=0.0
                      pveg(c,r,t)=0
                  endif
                  if(pveg(c,r,t).gt.lis%d%maxt) then
                      fgrd(c,r,t)=0.0           ! impose maxt cutoff
                      pveg(c,r,t)=0
                  endif
                  rsum=rsum+fgrd(c,r,t)
              enddo
!-----
! Renormalize veg fractions within a grid to 1
!-----
          if(rsum.gt.0.0) then
              do t=1,lis%p%nt
                  if(rsum.gt.0.0)fgrd(c,r,t)= fgrd(c,r,t)/rsum
              enddo

```

```

rsum=0.0
do t=1,lis%p%nt
    rsum=rsum+ fgrd(c,r,t) !recalculate rsum to check
enddo
tsum(c,r)=rsum

if(rsum.lt.0.9999.or.rsum.gt.1.0001)then !check renormalization
    write(*,*) 'error2 in vegetation tiles',rsum,c,r
endif
endif

enddo
enddo
deallocate(pveg)
call absoft_release_cache()

if ( masterproc ) then
    allocate(glbmask(lis%d%gnc, lis%d%gnr), stat=ierr)
else
    allocate(glbmask(1, 1), stat=ierr)
endif
call check_error(ierr,'Error allocating glbmask.',iam)
if ( npes > 1 ) then
    call MPI_GATHERV(mask,lis%d%gnc*lnr,MPI_REAL, &
        glbmask,recvcnts2,displs2,MPI_REAL, &
        0,MPI_COMM_WORLD, ierr)
else
    glbmask = mask
endif
deallocate(mask)
call absoft_release_cache()

if ( masterproc ) then
    allocate(glblat(lis%d%gnc, lis%d%gnr), stat=ierr)
else
    allocate(glblat(1, 1), stat=ierr)
endif
call check_error(ierr,'Error allocating glblat.',iam)
if ( npes > 1 ) then
    call MPI_GATHERV(lat,lis%d%gnc*lnr,MPI_REAL, &
        glblat,recvcnts2,displs2,MPI_REAL, &
        0,MPI_COMM_WORLD, ierr)
else
    glblat = lat
endif
deallocate(lat)
call absoft_release_cache()

```

```

if ( masterproc ) then
    allocate(glblon(lis%d%gnc, lis%d%gnr), stat=ierr)
else
    allocate(glblon(1, 1), stat=ierr)
endif
call check_error(ierr,'Error allocating glblon.',iam)
if ( npes > 1 ) then
    call MPI_GATHERV(lon,lis%d%gnc*lnr,MPI_REAL, &
                      glblon,recvcnts2,displs2,MPI_REAL, &
                      0,MPI_COMM_WORLD, ierr)
else
    glblon = lon
endif
deallocate(lon)
call absoft_release_cache()

if ( masterproc ) then
    allocate(glbfgrd(lis%d%gnc, lis%d%gnr, lis%p%nt), stat=ierr)
else
    allocate(glbfgrd(1, 1, 13), stat=ierr)
endif
call check_error(ierr,'Error allocating glbfgrd.',iam)
if ( npes > 1 ) then
    do t = 1, lis%p%nt
        call MPI_GATHERV(fgrd(:,:,t),lis%d%gnc*lnr,MPI_REAL, &
                          glbfgrd(:,:,t),recvcnts2,displs2,MPI_REAL, &
                          0,MPI_COMM_WORLD, ierr)
    enddo
else
    glbfgrd = fgrd
endif
deallocate(fgrd)
call absoft_release_cache()

if ( masterproc ) then
    allocate(glbtsum(lis%d%gnc, lis%d%gnr), stat=ierr)
else
    allocate(glbtsum(1, 1), stat=ierr)
endif
call check_error(ierr,'Error allocating glbtsum.',iam)
if ( npes > 1 ) then
    call MPI_GATHERV(tsum,lis%d%gnc*lnr,MPI_REAL, &
                      glbtsum,recvcnts2,displs2,MPI_REAL, &
                      0,MPI_COMM_WORLD, ierr)
else
    glbtsum = tsum
endif
deallocate(tsum)

```

```

call absoft_release_cache()

if ( masterproc ) then
  landnveg = 5
!    if (lis%p%koster .eq. 1) landnveg = 4
  lis%d%glbnch=0
  do t=1,lis%p%nt
    do r=1,lis%d%gnr
      do c=1,lis%d%gnc
        if(glbmask(c,r).gt.0.99.and. &
           glbmask(c,r).lt.3.01)then
          if(glbfrd(c,r,t).gt.0.0)then
            lis%d%glbnch=lis%d%glbnch+1
          endif
          if(glbtsum(c,r).eq.0.0.and.t.eq.landnveg)then
            lis%d%glbnch=lis%d%glbnch+1
          endif
        endif
      enddo
    enddo
  enddo
  print*, 'DBG: maketiles -- glbnch',lis%d%glbnch,' (',iam,')
  allocate(tile(lis%d%glbnch))

  lis%d%glbngrid=0
  do r=1,lis%d%gnr
    do c=1,lis%d%gnc
      if(glbmask(c,r).gt.0.99 .and. &
         glbmask(c,r).lt.3.01) then
        lis%d%glbngrid=lis%d%glbngrid+1
      endif
    enddo
  enddo
  count = 1
  print*, 'DBG: maketiles1 -- glbnch',lis%d%glbnch,' (',iam,')
  allocate(grid(lis%d%glbngrid))
  allocate(glbgindex(lis%d%gnc, lis%d%gnr))
  print*, 'DBG: maketiles2 -- glbnch',lis%d%glbnch,' (',iam,')
  do r=1,lis%d%gnr
    do c=1,lis%d%gnc
      glbgindex(c,r) = -1
      if(glbmask(c,r).gt.0.99 .and. &
         glbmask(c,r).lt.3.01) then
        grid(count)%lat = glblat(c,r)
        grid(count)%lon = glblon(c,r)
        grid(count)%fgrd = glbfgrd(c,r,:)
        glbgindex(c,r) = count
    enddo
  enddo
end

```

```

        count = count+1
    endif
enddo
enddo
deallocate(glblat,stat=ierr)
call absoft_release_cache()
call check_error(ierr,'Error allocating glblat.',iam)
deallocate(glblon, stat=ierr)
call check_error(ierr,'Error allocating glblon.',iam)
print*, 'DBG: maketiles3 -- glbnch',lis%d%glbnch,' (',iam,')'
count = 0
do r=1,lis%d%gnr
    do c=1,lis%d%gnc
        do t=1,lis%p%nt
            if(glbmask(c,r).gt.0.99.and. &
               glbmask(c,r).lt.3.01)then
                if(glbfgrd(c,r,t).gt.0.0)then
                    count = count+1
                    tile(count)%row=r
                    tile(count)%col=c
                    tile(count)%index = glbgindex(c,r)
                    tile(count)%vegt=t
                    tile(count)%fgrd=glbfgrd(c,r,t)
                endif
                if(glbtsum(c,r).eq.0.0.and.t.eq.landnveg)then
                    count=count+1
                    tile(count)%row=r
                    tile(count)%col=c
                    tile(count)%index = glbgindex(c,r)
                    tile(count)%vegt=t
                    tile(count)%fgrd=1.0
                endif
            endif
        enddo
    enddo
enddo
print*, 'DBG: maketiles4 -- glbnch',lis%d%glbnch,' (',iam,')'
deallocate(glbmask, stat=ierr)
call check_error(ierr,'Error allocating glbmask',iam)
deallocate(glbfgrd, stat=ierr)
call check_error(ierr,'Error allocating glbfgrd',iam)
deallocate(glbtsum, stat=ierr)
call check_error(ierr,'Error allocating glbtsum.',iam)
call absoft_release_cache()
WRITE(*,*) 'MSG: maketiles -- Size of Tile Dimension:',NCHP, &
' (',iam,')'
WRITE(*,*) 'MSG: maketiles -- Actual Number of Tiles:', &
LIS%D%GLBNCH,' (',iam,')'

```

```
WRITE(*,*)

WRITE(*,*) 'MSG: maketiles -- Size of Grid Dimension:', &
lis%d%glbngrid,' (',iam,')'
WRITE(*,*) 
WRITE(79,*) 'MSG: maketiles -- Size of Tile Dimension:',NCHP, &
' (',iam,')'
WRITE(79,*) 'MSG: maketiles -- Actual Number of Tiles:', &
LIS%D%GLBNCH,' (',iam,')'
WRITE(79,*)

endif
print*, 'MSG: maketiles -- done', ' (',iam,')'

#endif
return
```